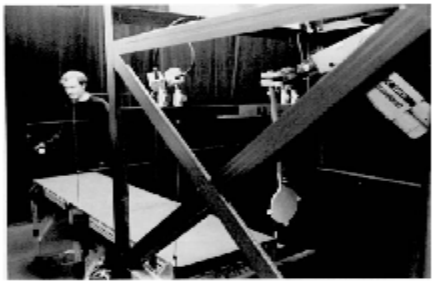
CS 188: Artificial Intelligence
Spring 2007

Lecture 2: Agents
1/18/2007

Srini Narayanan – ICSI and UC Berkeley
Many slides from Dan Klein, Mitch Marcus

Administrivia

- Reminder:
  - Sections and Office hours start next week.
  - Schedules posted soon.
- Accommodation issues
- Assignment 0 part 1 is the tutorial (not graded) which should be up. Part 1 will be up by Friday and is a written assignment covering the first two weeks of lecture. Due 11:59 PM on 1/30.
- Course workload curve



Today

- Agents and Environments
- Reflex Agents
- Environment Types
- Problem-Solving Agents

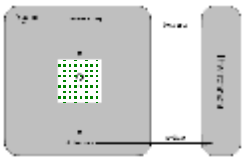
Agents and Environments

- Agents include:
  - Humans
  - Robots
  - Softbots
  - Thermostats
  - …
- The agent function maps from percept histories to actions:

$$\mathcal{P}^* \rightarrow \mathcal{A}$$

- An agent program running on the physical architecture to produce the agent function.

The line between agent and environment depends on the level of abstraction.

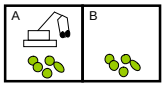


Always think of the environment as a black box, completely external to the agent – even if it's simulated by local code.

Agents

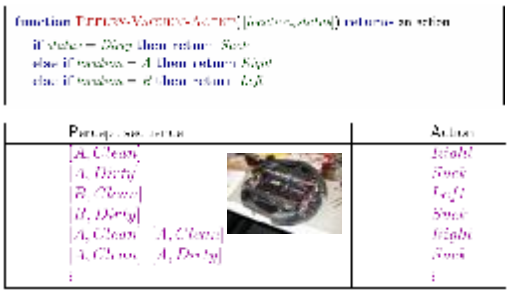
- An *agent* is anything that can be viewed as
  - *perceiving* its *environment* through *sensors* and
  - *acting* upon that environment through *actuators*
- Human agent:
  - Sensors: eyes, ears, ...
  - Actuators: hands, legs, mouth, …
- Robotic agent:
  - Sensors: cameras and infrared range finders
  - Actuators: various motors

Example: A Vacuum-cleaner agent



- Percepts:location and contents, e.g. ***[A, dirty]***
  - *(Idealization: locations are discrete)*
- Actions: ***LEFT, RIGHT, SUCK, NOP***

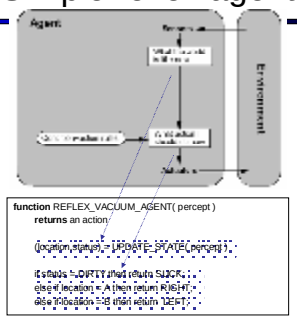
A Reflex Vacuum-Cleaner



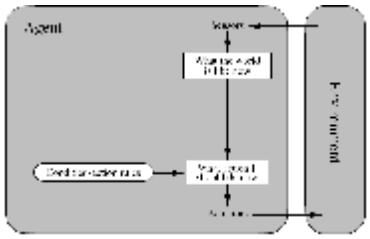
Python code for agent

```
loc_A, loc_B = (0, 0), (1, 0)
# The two locations for the Vacuum world class

ReflexVacuumAgent(Agent):
"A reflex agent for the two-state vacuum environment. [Fig. 2.8]"

def __init__(self):
        Agent.__init__(self)
        def program((location, status)):
                if status == 'Dirty': return 'Suck'
                elif location == loc_A: return 'Right'
                elif location == loc_B: return 'Left'
        self.program = program
```

Simple reflex agent



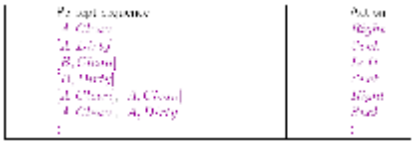
Simple Reflex Agents



- Does this ever make sense as a design?

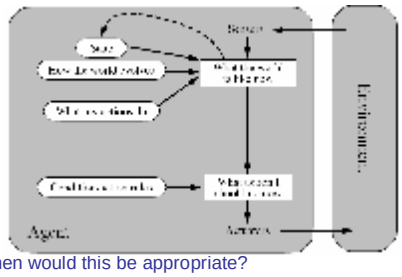
Table-Lookup Agents?

- Complete map from percept (histories) to actions



- Drawbacks:
  - Huge table!
  - No autonomy
  - Even with learning, need a long time to learn the table entries
- How would you build a spam filter agent?
- Most agent programs produce complex behaviors from compact specifications

Reflex Agents with State



- When would this be appropriate?
- How do we update state?

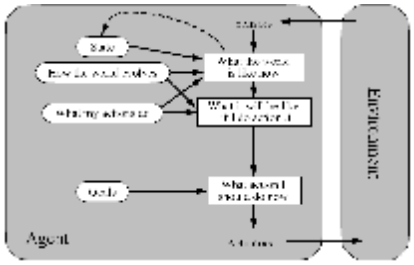
Rationality

- A fixed performance measure evaluates the environment sequence
  - One point per clean square at time T?
  - One point per clean square per time step, minus one per move?
  - Penalize for > k dirty squares?
- Reward should indicate success, not steps to success
- A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date
- Rational ≠ omniscient: percepts may not supply all information
- Rational ≠ clairvoyant: action outcomes may not be as expected
- Hence, rational ≠ successful

Rationality and Goals

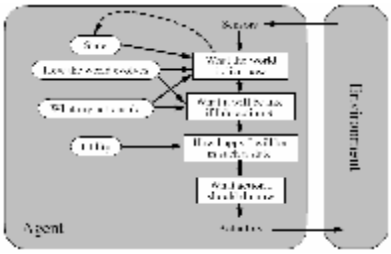
- Let's say we have a game:
  - Flip a biased coin (probability of heads is h)
  - Tails = loose $1
  - Heads = win $1
- What is the expected winnings?
  - (1)(h) + (-1)(1-h) = 2h - 1
- Rational to play?
  - What if performance measure is total money?
  - Why might a human play this game at expected loss?

Goal-Based Agents



- These agents usually first find plans then execute them.

Utility-Based Agents



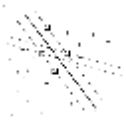
- How is this different from a goal-based agent?

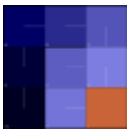
More Rationality

- Remember: rationality depends on:
  - Performance measure
  - Agent's (prior) knowledge
  - Agent's percepts to date
  - Available actions
- Is it rational to inspect the street before crossing?
- Is it rational to try new things?
- Is it rational to update beliefs?
- Is it rational to construct conditional plans in advance?
- Rationality gives rise to: exploration, learning, autonomy

The Road Not (Yet) Taken

- At this point we could go directly into:
  - Empirical risk minimization *(statistical classification)*
  - Expected return maximization *(reinforcement learning)*
- These are mathematical approaches that let us derive algorithms for rational action for reflex agents under nasty, realistic, uncertain conditions
- But we'll have to wait until week 5, when we have enough probability to work it all through
- Instead, we'll first consider more general goal-based agents, but under nice, deterministic conditions





Task environment

- To design a rational agent we need to specify a *task environment*
  - a problem specification for which the agent is a solution
- *PEAS: to* **specify a task environment**
  - ***P***erformance measure
  - ***E***nvironment
  - ***A***ctuators
  - ***S***ensors

PEAS: Specifying an automated taxi driver

***P*erformance measure**:
- ?

***E*nvironment**:
- ?

***A*ctuators**:
- ?

***S*ensors**:
- ?

PEAS: Specifying an automated taxi driver

***P*erformance measure**:
- safety, speed, legal, comfortable, maximize profits

***E*nvironment**:
- ?

***A*ctuators**:
- ?

***S*ensors**:
- ?

PEAS: Specifying an automated taxi driver

***P*erformance measure**:
- safe, fast, legal, comfortable, maximize profits

***E*nvironment**:
- roads, other traffic, pedestrians, customers

***A*ctuators**:
- ?

***S*ensors**:
- ?

PEAS: Specifying an automated taxi driver

***P*erformance measure**:
- safe, fast, legal, comfortable, maximize profits

***E*nvironment**:
- roads, other traffic, pedestrians, customers

***A*ctuators**:
- steering, accelerator, brake, signal,

***S*ensors**:
- ?

PEAS: Specifying an automated taxi driver

***P*erformance measure**:
- § safe, fast, legal, comfortable, maximize profits

***E*nvironment**:
- § roads, other traffic, pedestrians, customers

***A*ctuators**:
- § steering, accelerator, brake, signal, horn

***S*ensors**:
- § cameras, sonar, speedometer, GPS

PEAS: Internet Shopping Agent

§ Specifications:

- § Performance measure: price, quality, appropriateness, efficiency
- § Environment: current and future WWW sites, vendors, shippers
- § Actuators: display to user, follow URL, fill in form
- § Sensors: HTML pages (text, graphics, scripts)

PEAS: Spam Filtering Agent

§ Specifications:

- § Performance measure: spam block, false positives, false negatives
- § Environment: email client or server
- § Actuators: mark as spam, transfer messages
- § Sensors: emails (possibly across users), traffic, etc.

Environment Simplifications

§ Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.

§ Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent.

§ Episodic (vs. sequential): The agent's experience is divided into independent atomic "episodes" (each episode consists of the agent perceiving and then performing a single action)

Environment Simplifications

§ Static (vs. dynamic): The environment is unchanged while an agent is deliberating.

§ Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.

§ Single agent (vs. multi-agent): An agent operating by itself in an environment.

§ What's the real world like?

Environment Types

	Crossword puzzle	Back-gammon	Internet Shopping	Taxi
Observable	✓	✓	✗	✗
Deterministic	✓	✗	?	✗
Episodic	✗	✗	✗	✗
Static	✓	✓	?	✗
Discrete	✓	✓	✓	✗
Single-Agent	✓	✗	✓	✗

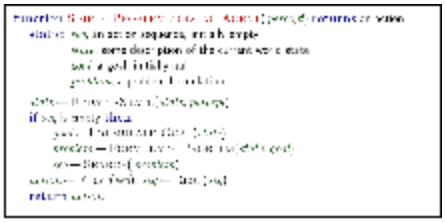
§ The environment type largely determines the agent design

§ The real world is partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Agent Types

- Four basic types in order of increasing generality:
  - Simple reflex agents (classification) – Week 6-8
  - Reflex agents with state (reinforcement learning) – Week 8-10
  - Goal-based agents (Problem solving with search and planning)) Week 2-4
  - Utility-based agents (probabilistic reasoning) Week 4-6
- All these can be turned into learning agents

Problem-Solving Agents



- This offline problem solving!
- Solution is executed "eyes closed."
- When will offline solutions work? Fail?

Example: Romania

Example: Romania

- Setup
  - On vacation in Romania; currently in Arad
  - Flight leaves tomorrow from Bucharest
- Formulate problem:
  - States: being in various cities
  - Actions: drive between adjacent cities
- Define goal:
  - Being in Bucharest
- Find a solution:
  - Sequence of actions, e.g. [Arad → Sibiu, Sibiu → Fagaras, …]

Problem Formulation: Types

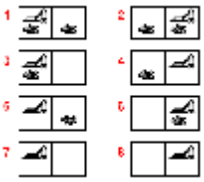
- Deterministic, fully observable à single-state problem
  - Agent knows exactly which state it will be in; solution is a sequence, can solve offline using model of environment
- Non-observable à sensorless problem (conformant problem)
  - Agent may have no idea where it is; solution is a sequence
- Nondeterministic and/or partially observable à contingency problem
  - Percepts provide new information about current state
  - Often first priority is gathering information or coercing environment
  - Often interleave search, execution
  - Cannot solve offline
- Unknown state space à exploration problem

Single State Problems

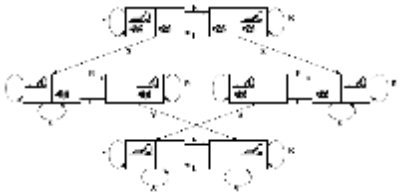
- A search problem is defined by four items:
  - Initial state: e.g. Arad
  - Successor function S(x) = set of action–state pairs:
    e.g., S(Arad) = {<Arad → Zerind, Zerind>, … }
  - Goal test, can be
    - explicit, e.g., x = Bucharest
    - implicit, e.g., Checkmate(x)
  - Path cost (additive)
    - e.g., sum of distances, number of actions executed, etc.
    - c(x,a,y) is the step cost, assumed to be ≥ 0
- A solution is a sequence of actions leading from the initial state to a goal state
- Problem formulations are almost always abstractions and simplifications

Example: Vacuum World

- States?
- Goal?
- Single-State: Start in 5.
  - Solution?
  - [Right, Suck]
- Sensorless: Start in {1…8}
  - Solution?
  - [Right, Suck, Left, Suck]

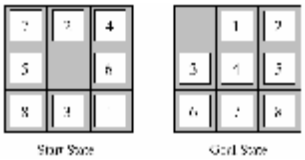


Example: Vacuum World



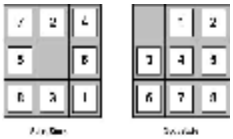
- Can represent problem as a graph
  - Nodes are states
  - Arcs are actions

Example: 8-Puzzle



- What are the states?
- What are the actions?
- What states can I reach from the start state?
- What should the costs be?

Example: 8-puzzle



- States?? Integer location of each tile
- Initial state?? Any state can be initial
- Actions?? {*Left*, *Right*, *Up*, *Down*}
- Goal test?? Check if goal configuration is reached
- Path cost?? Number of actions to reach goal

Example: Missionaries & Cannibals

Three missionaries and three cannibals come to a river. A rowboat that seats two is available. If the cannibals ever outnumber the missionaries on either bank of the river, the missionaries will be eaten.

How shall they cross the river?

Formulation: Missionaries & Cannibals

- *How to formalize:*
  - *Initial state*: all M, all C, and boat on one bank
  - *Goal test*: True if all M, all C, and boat on other bank
  - *Operators*: ??
  - *Cost*: ??

Remember:

- *Representation:*
  - *States: Which properties matter & how to represent*
  - *Operators: Which actions are possible & how to represent*
  - *Path Cost: Deciding which action is next*

Missionaries and Cannibals

States: (ML, CL, BL)
Initial 331 Goal 000

Operators:

Travel Across	*Travel Back*
-101	101
-201	201
-011	011
-021	021
-111	111

Solution 1: 331-310-321-300-311-110-221-020-031-010-021-000
Solution 2: 331-220-321-300-311-110-221-020-031-010-021-000

Summary

- Agents interact with environments through actuators and sensors
  - The agent function describes what the agent does in all circumstances
  - The agent program calculates the agent function
  - The performance measure evaluates the environment sequence
- A perfectly rational agent maximizes expected performance
- PEAS descriptions define task environments
- Environments are categorized along several dimensions:
  - Observable? Deterministic? Episodic? Static? Discrete? Single-agent?
- Problem-solving agents make a plan, then execute it
- State space encodings of problems